

**CIRCULATION COPY**  
**SUBJECT TO RECALL**  
**IN TWO WEEKS**

UCRL-84095  
PREPRINT

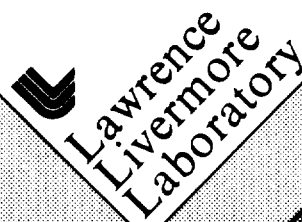
COMPARING THE FLOATING POINT SYSTEMS, INC. AP-190L  
TO REPRESENTATIVE SCIENTIFIC COMPUTERS:  
SOME BENCHMARK RESULTS

Thomas A. Brengle and Neil Maron

This paper was prepared for presentation at the  
1980 FOURTH ANNUAL FPS USER'S CONFERENCE  
San Francisco, CA.  
April 28, 1980

March 27, 1980

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

The logo for Lawrence Livermore Laboratory, featuring a stylized 'L' symbol and the text 'Lawrence Livermore Laboratory' arranged in a triangular shape.

Lawrence  
Livermore  
Laboratory

#### DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

COMPARING THE FLOATING POINT SYSTEMS, INC. AP-190L  
TO REPRESENTATIVE SCIENTIFIC COMPUTERS:  
SOME BENCHMARK RESULTS\*

Thomas A. Brengle and Neil Maron  
Lawrence Livermore Laboratory, University of California,  
Livermore, California 94550

ABSTRACT

In this paper we present the results of comparative timing tests made by running a typical FORTRAN physics simulation code on the following machines:

1. DEC PDP-10 with KI processor.
2. DEC PDP-10, KI processor, and FPS AP-190L.
3. CDC 7600.
4. CRAY-1.

Factors, such as DMA overhead, code size for the AP-190L, and the relative utilization of floating point functional units for the different machines, are discussed.

INTRODUCTION

Researchers in the Magnetic Fusion Energy Program at the Lawrence Livermore Laboratory regularly use FORTRAN codes of all sizes to help provide solutions to many types of engineering and physics problems. Even though the

---

\*Work performed under the auspices of the U. S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48.

computational resources available include a Digital Equipment PDP-10 with KI processor, a Control Data 7600, and a CRAY-1, it is often the case that the elapsed time between the start of a code and the output of the results at investigator's terminal is quite long. This delay is due to many factors; not the least being that these machines operate in a time-sharing environment.

Many codes are too large to be run on any machine but a CDC 7600 or CRAY-1. However, there are several codes, although not necessarily large, that still require a great deal of computational power. In an effort to reduce the turnaround time of these codes, we connected a Floating Point Systems AP-190L to act as a slave to our DEC PDP-10. It was installed in June 1978 with the complete software package as available at that time.

Timing tests verified that the AP-190L hardware was indeed fast. However, without a FORTRAN cross-compiler for the AP-190L, conversion of existing codes would be slow and arduous. In 1979, the installation of software release 79.1, which included AP FORTRAN,<sup>1</sup> promised to change that situation.

#### THE BENCHMARK

In order to make some comparative tests, we decided to use, as a benchmark, a code which had been developed for use on the PDP-10 and which had reached a plateau in its development effort. This code, called MAGIC2 by its authors,<sup>2</sup> is a one-dimensional cylindrically-symmetric quasi-neutral magneto-inductive particle code, with electromagnetic fields varying only as a function of radius. The code includes only radius, radial velocity, azimuthal velocity, and azimuthal canonical momentum as degrees of freedom for the

particles. While it is not important to this discussion that the physics of the simulation be understood, a simplified verbal flow chart of this code is as follows:

1. Read in parameters and initialize output files.
2. Enter the simulation particles and their physical characteristics.
3. Accumulate the current due to the initial velocities of the particles.
4. Repeat the following loop several times:
  - a. Solve for new electric and magnetic fields as induced by particle currents.
  - b. Move each particle an incremental amount according to the new fields, and accumulate the new currents due to their velocities. Also, provide for the particles which are moved out of the system.
  - c. Occasionally sample the physical quantities of interest in the system, for instance: local field values, particle velocities, etc.
  - d. Go back to (a).
5. When the loop is finished, perform some diagnostics on the system, such as determination of local particle densities, energy densities, etc.
6. If enough simulation time has not elapsed, go back and repeat step (4).
7. Otherwise, do the historical summaries of sampled information and terminate the run.

The majority of the computation was within step (4). Moving of the particles in step (4b) actually required 70 to 90 percent of the looptime.

#### PROCEDURE

The comparative test procedure was as follows:

1. Modify the source, as necessary, to allow error-free compilation on the given machine.
2. Compile and load the code.
3. Execute code for a typical problem.

While step (3) of the test procedure gave a measure of the hardware speed, it was felt that there should also be a comparison of the time required to set up the executable code, as this could be important when code development or debugging might be in progress.

#### COMPARISON OF SETUP TIMES

In each case the conversion of the source from the PDP-10 to the CDC 7600 and CRAY-1 required about the same amount of time. Approximately four hours of work was needed to make the source compatible with the two resident FORTRAN compilers: CHAT on the CDC 7600 and CFT on the CRAY-1. This included the time required to make several runs of the compilers in order to deal with the errors arising from slight compiler differences. The run time to do a compile and load was about 45 seconds on the CDC 7600 and about 2 seconds on the CRAY-1.

However, the conversion to the AP required more than 2 days, due mainly to two factors. The first was that AP FORTRAN and APLOAD<sup>3</sup> were written in host FORTRAN and ran on the host machine. They were very slow. The run time to do a compile and load was about 1 hour. The second was due to the relative small size of the program memory in the AP-190L. When compiling and loading code for the AP-190L, it was difficult to know what the final sizes of the modules would be. Although modules that were too large could be broken up into two or more overlays, it was difficult to determine a convenient size until several passes through the compile and load procedure had been made.

When overlays were introduced, data memory management became a problem. Since overlays occupy twice as much data memory as program memory, the amount of data memory available to the code for data storage was reduced significantly. In the final configuration of the AP-190L version of the code, the major loop was the only piece of code running in the AP-190L. The rest of the code was not repetitive and was primarily input/output operations which the AP-190L FORTRAN does not support, and which the AP-190L was not able to initiate in our system's configuration. Within the loop, each step was assigned its own overlay, resulting in a driver which was always resident in program memory, and three overlays.

It should be noted at this point that vectorization was difficult within the structure of the benchmark program and could not be done at all by an automatically vectorizing compiler like CFT on CRAY-1.

#### A TYPICAL PROBLEM

The two benchmark code parameters, which primarily determined the length of time that the problem required to be completed, were: the number of

simulation particles used and the number of simulation time steps that the code was allowed to run. The run time required increased linearly as either parameter was increased. As a representative case, we chose to use 2000 simulation particles, and to let the simulation run for 1024 time steps. We knew from past experience that this was a typical setup, and was too long to attempt to run on the PDP-10 by itself.

We also knew from past experience<sup>4</sup> that we would have to consider the overhead incurred by the APEX calls to the PDP-10 operating system, and the overhead due to the DMA transfers of data to and from the AP-190L. This overhead was minimized by allowing the simulation to run 128 time steps for each AP-190L run call. This permitted the AP-190L to run for several seconds each time the loop was executed.

## RESULTS

The results of the test runs are tabulated in Table I.

Table I. Benchmark Results (in seconds)

	DEC PDP-10	FPS <sup>c)</sup> AP-190L	CDC 7600	CRAY-1
MIPS <sup>a)</sup>	1	18	36	80
Theoretical MFLOPS <sup>b)</sup>	0.25	12	54-56	160-240
Elapsed time	5640	564	375	271
CPU	2559	276	99	50
MFLOPS	0.14	1.3	3.5	7.0
Realized MFLOPS/Theoretical MFLOPS	0.56	0.11	0.06	0.04
Megabucks/Realized MFLOPS	3.6	0.08	1.4	1.1

### NOTES:

- MIPS: Million instructions executed per second.
- MFLOPS: Million floating-point operations per second.
- AP109L CPU time was determined by counting cycles.



### SOME OBSERVATIONS

From the table, it can be seen that the AP-190L was able to improve the turnaround (elapsed) time by a factor of 10 over what was possible with the PDP-10 by itself. In addition to this, we observed that the overhead incurred by the host was less than 5 percent, which we felt to be an acceptable figure. The turnaround time was almost within a factor of 2 of that for the CRAY-1, which we also felt was very good. Of course, the CRAY-1 was operating in a time-sharing situation, so this implies more that the CRAY-1 was heavily loaded than it does that the AP-190L has half the computational power of a CRAY-1. This can be seen by looking at the CPU time which indicates a ratio of close to 5:1.

Using the ratio of realized MFLOPS to theoretical MFLOPS as a measure of the floating point functional unit utilization, we see that relatively little use of the functional units was made, and that neither more functional units nor faster ones could be expected to improve the performance of this type of code. This was largely due to the very scalar nature of the code.

The ratios of megabucks to realized MFLOPS clearly shows that if the AP-190L was not the fastest benchmark routine run, it certainly was the most cost effective.

### CONCLUSIONS

Several conclusions can be reached as a result of these comparative timing tests.

First, while the AP-190L hardware is very fast, the AP-190L code development software is not. This means that the AP-190L will prove to be a cost effective resource, if and only if, it is primarily used in a production environment where code modifications are kept to a minimum.

Second, the user must constantly be aware of the host's overhead when making use of the AP-190L. Calls to the host operating system must be minimized, and DMA transfers must be minimized as often as possible.

Third, the run time in the AP-190L should be maximized so the overhead becomes comparatively small.

Fourth, the AP-190L is capable of efficiently running only relatively small programs, even with the use of overlays. Overlays use twice as much data memory as program memory and this tends to use up data memory quickly as overlays are introduced.

And lastly, if all of the above are taken into consideration for a particular problem program, it is quite possible for the AP-190L to be a highly acceptable substitute for one of the much larger machines.

## REFERENCES

1. Array Processor Fortran Reference Manual, Floating Point Systems, Inc., Publication No. FPS 860-7408-000, November 1978.
2. T. A. Brengle, B. I. Cohen, MAGIC: A One-Dimensional Magneto-Inductive Particle Code, University of California UCID-17795, Rev. 1, July 18, 1978.
3. APLOAD Reference Manual, Floating Point Systems, Inc., Publication No. FPS 860-7410-000, January 1979.
4. Neil Maron and George G. Sutherland, AP190-L and PDP-K110: A Hardware/Software Measurement Report, University of California UCRL-82652, May 25, 1979 (contained in 79 UG 3/FPS, "Record of 1979 User's Group Meeting").

## NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.